AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Cancelled).

2. (Cancelled).

3. (Cancelled).

4. (Cancelled).

5. (Currently Amended) A computer executable method of managing information about where to begin recovery after a failure of one or more nodes of a multiple-node system, the method comprising the steps of:

in a particular node of [[a]] the multiple-node system, maintaining both

a single-failure queue ~~that indicates~~ indicating where within a recovery log to begin recovery after a failure of said node, and

a multiple-failure queue ~~that indicates~~ indicating where within said recovery log to begin recovery after a failure of said node and one or more other nodes in said multiple-node system;

in response to a dirty data item being written to persistent storage, removing an entry for said data item from both said single-failure queue and said multiple-failure queue; and

in response to a dirty data item being sent to another node of said multiple-node system without first being written to persistent storage, removing an entry for said data item from said single-failure queue without removing the entry for said data item from said multiple-failure queue.

6. (Original) The method of Claim 5 further comprising the step of sending the dirty data item to another node to allow removal of the entry from said single-failure queue without the other node requesting the dirty data item.

7. (Original) The method of Claim 5 further comprising the steps of:

after a single node failure, applying said recovery log beginning at a position in said recovery log associated with the single-failure queue; and

after a multiple node failure, applying said recovery log beginning at a position in said recovery log associated with the multiple-failure queue.

8. (Original) The method of Claim 5 wherein:

said single-failure queue and said multiple-failure queue are implemented by a single combined queue; and

the step of removing an entry for said data item from said single-failure queue without removing the entry for said data item from said multiple-failure queue includes marking an entry for said data item in said combined queue without removing the entry for said data item from said combined queue.

9. (Original) The method of Claim 5 wherein said single-failure queue and said multiple-failure queue are implemented as two separate queues.

10. (Currently Amended) A computer executable method for recovering after a failure of one or more nodes of a multiple-node system, the method comprising the steps of:

determining whether the failure involves only one node; and

if the failure involves only said one node, then performing recovery by applying a recovery log of said node beginning at a first point in the recovery log; and

if the failure involves one or more nodes in addition to said one node, then

performing recovery by applying said recovery log of said node beginning at a second point in the recovery log;

wherein said first point is different from said second point.

11. (Original) The method of Claim 10 wherein:

the first point is determined, at least in part, by which data items that were dirtied by said node reside in caches in other nodes; and

the second point is determined, at least in part, by which data items that were dirtied by said node have been persistently stored.

12. (Currently Amended) A computer executable method for recovering after a failure of one or more nodes of a multiple-node system, the method comprising the steps of:

if it is unclear whether a particular version of a data item has been written to disk, then performing the steps of

without attempting to recover said data item, marking dirtied cached versions of said data item that would have been covered if said particular version was written to disk;

when a request is made to write one of said dirtied cached versions to disk, determining which version of said data item is already on disk; and

if said particular version of said data item is already on disk, then not writing said one of said dirtied cached versions to disk.

13. (Original) The method of Claim 12 further comprising the step of, if said particular version of said data item is not already on disk, then recovering said data item.

14. (Original) The method of Claim 12 further comprising the step of, if said particular version of said data item is already on disk, then informing nodes that contain said

dirtied cached versions of the data item that said dirtied cached versions are covered by a write-to-disk operation.

15. (Cancelled).

16. (Currently Amended) A computer executable method for recovering a current version of a data item after a failure of one or more nodes in a system that includes multiple caches, the method comprising the steps of:

modifying the data item in a first node of said multiple caches to create a modified data item;

sending the modified data item from said first node to a second node of said multiple caches without durably storing the modified data item from said first node to persistent storage;

after said modified data item has been sent from said first node to said second node and before said data item in said first node has been covered by a write-to-disk operation, discarding said data item in said first node;

after said failure, reconstructing the current version of said data item by applying changes to the data item on persistent storage based on merged redo logs associated with all of said multiple caches;

maintaining, for each of said multiple caches, a globally-dirty checkpoint queue and a locally-dirty checkpoint queue;

wherein the globally-dirty data items associated with entries in the globally-dirty checkpoint queue are not retained until covered by write-to-disk operations;

determining, for each cache, a checkpoint based on a lower of a first-dirtied time of the entry at the head of the locally-dirty checkpoint queue and the first-dirtied time of the entry at the head of the globally-dirty checkpoint queue; and

after said failure, determining where to begin processing the redo log associated with each cache based on the checkpoint determined for said cache.

17. (Currently Amended) A computer executable method for recovering a current version of a data item after a failure of one or more nodes in a system that includes multiple caches, the method comprising the steps of:

modifying the data item in a first node of said multiple caches to create a modified data item;

sending the modified data item from said first node to a second node of said multiple caches without durably storing the modified data item from said first node to persistent storage;

after said modified data item has been sent from said first node to said second node and before said data item in said first node has been covered by a write-to-disk operation, discarding said data item in said first node; and

after said failure, reconstructing the current version of said data item by applying changes to the data item on persistent storage based on merged redo logs associated with all of said multiple caches;

maintaining, for each of said multiple caches, a globally-dirty checkpoint queue and a locally-dirty checkpoint queue;

wherein the globally-dirty data items associated with entries in the globally-dirty checkpoint queue are not retained until covered by write-to-disk operations;

maintaining, for each cache, a first checkpoint record for the locally-dirty checkpoint queue that indicates a first time, where all changes made to data items that are presently dirty in the cache prior to the first time have been recorded on a version of the data item that is on persistent storage;

maintaining, for each cache, a second checkpoint record for the globally-dirty checkpoint queue, wherein the second checkpoint record includes a list of data items that were once dirtied in the cache but have since been transferred out and not written to persistent storage; and

after said failure, determining where to begin processing the redo log associated with each cache based on the first checkpoint record and said second checkpoint record for said cache.

18. (Original) The method of Claim 17 wherein the step of processing the redo log comprises the steps of:

determining a starting position for scanning the redo log based on a lesser of a position in the redo log as determined by the first checkpoint record and the positions in the log as determined by the earliest change made to the list of the data items in the second checkpoint record; and

during recovery, for the portion of the redo log between the position indicated by the global checkpoint record to the position indicated by the local checkpoint record, considering for potential redo only those log records that correspond to the data items identified in the global checkpoint record.

19. (Cancelled).

20. (Cancelled).

21. (Cancelled).

22. (Cancelled).

23. (Currently Amended) A computer-readable medium carrying instructions for managing information about where to begin recovery after a failure <u>of one or more</u>

nodes of a multiple-node system, the instructions comprising instructions for performing the steps of:

in a particular node of a multiple-node system, maintaining both

a single-failure queue that indicates where within a recovery log to begin recovery after a failure of said node, and

a multiple-failure queue that indicates where within said recovery log to begin recovery after a failure of said node and one or more other nodes in said multiple-node system;

in response to a dirty data item being written to persistent storage, removing an entry for said data item from both said single-failure queue and said multiple-failure queue; and

in response to a dirty data item being sent to another node of said multiple-node system without first being written to persistent storage, removing an entry for said data item from said single-failure queue without removing the entry for said data item from said multiple-failure queue.

24. (Original) The computer-readable medium of Claim 23 further comprising instructions for performing the step of sending the dirty data item to another node to allow removal of the entry from said single-failure queue without the other node requesting the dirty data item.

25. (Original) The computer-readable medium of Claim 23 further comprising instructions for performing the steps of:

after a single node failure, applying said recovery log beginning at a position in said recovery log associated with the single-failure queue; and

after a multiple node failure, applying said recovery log beginning at a position in said recovery log associated with the multiple-failure queue.

26. (Original) The computer-readable medium of Claim 23 wherein:

said single-failure queue and said multiple-failure queue are implemented by a single combined queue; and

the step of removing an entry for said data item from said single-failure queue without removing the entry for said data item from said multiple-failure queue includes marking an entry for said data item in said combined queue without removing the entry for said data item from said combined queue.

27. (Original) The computer-readable medium of Claim 23 wherein said single-failure queue and said multiple-failure queue are implemented as two separate queues.

28. (Currently Amended) A computer-readable medium carrying instructions for recovering after a failure of one or more nodes of a multiple-node system, the instructions comprising instructions for performing the steps of:

determining whether the failure involves only one node; and

if the failure involves only said one node, then performing recovery by applying a recovery log of said node beginning at a first point in the recovery log; and

if the failure involves one or more nodes in addition to said one node, then performing recovery by applying said recovery log of said node beginning at a second point in the recovery log;

wherein said first point is different from said second point.

29. (Original) The computer-readable medium of Claim 28 wherein:

the first point is determined, at least in part, by which data items that were dirtied by said node reside in caches in other nodes; and

the second point is determined, at least in part, by which data items that were dirtied by said node have been persistently stored.

30. (Currently Amended) A computer-readable medium carrying instructions for recovering after a failure of one or more nodes of a multiple-node system, the instructions comprising instructions for performing the steps of:

if it is unclear whether a particular version of a data item has been written to disk, then performing the steps of

without attempting to recover said data item, marking dirtied cached versions of said data item that would have been covered if said particular version was written to disk;

when a request is made to write one of said dirtied cached versions to disk, determining which version of said data item is already on disk; and

if said particular version of said data item is already on disk, then not writing said one of said dirtied cached versions to disk.

31. (Original) The computer-readable medium of Claim 30 further comprising instructions for performing the step of, if said particular version of said data item is not already on disk, then recovering said data item.

32. (Original) The computer-readable medium of Claim 30 further comprising instructions for performing the step of, if said particular version of said data item is already on disk, then informing nodes ~~that contain~~ containing said dirtied cached versions of the data item that said dirtied cached versions are covered by a write-to-disk operation.

33. (Cancelled).

34. (Currently Amended) A computer-readable medium carrying instructions for recovering a current version of a data item after a failure of one or more nodes in a system that includes multiple caches, the instructions comprising instructions for performing the steps of:

modifying the data item in a first node of said multiple caches to create a modified data item;

sending the modified data item from said first node to a second node of said multiple caches without durably storing the modified data item from said first node to persistent storage;

after said modified data item has been sent from said first node to said second node and before said data item in said first node has been covered by a write-to-disk operation, discarding said data item in said first node;

after said failure, reconstructing the current version of said data item by applying changes to the data item on persistent storage based on merged redo logs associated with all of said multiple caches;

maintaining, for each of said multiple caches, a globally-dirty checkpoint queue and a locally-dirty checkpoint queue;

wherein the globally-dirty data items associated with entries in the globally-dirty checkpoint queue are not retained until covered by write-to-disk operations;

determining, for each cache, a checkpoint based on a lower of a first-dirtied time of the entry at the head of the locally-dirty checkpoint queue and the first-dirtied time of the entry at the head of the globally-dirty checkpoint queue; and

after said failure, determining where to begin processing the redo log associated with each cache based on the checkpoint determined for said cache.

35. (Currently Amended) A computer-readable medium carrying instructions for recovering a current version of a data item after a failure of one or more nodes in a system that includes multiple caches, the instructions comprising instructions for performing the steps of:

modifying the data item in a first node of said multiple caches to create a modified data item;

sending the modified data item from said first node to a second node of said multiple caches without durably storing the modified data item from said first node to persistent storage;

after said modified data item has been sent from said first node to said second node and before said data item in said first node has been covered by a write-to-disk operation, discarding said data item in said first node;

after said failure, reconstructing the current version of said data item by applying changes to the data item on persistent storage based on merged redo logs associated with all of said multiple caches;

maintaining, for each of said multiple caches, a globally-dirty checkpoint queue and a locally-dirty checkpoint queue;

wherein the globally-dirty data items associated with entries in the globally-dirty checkpoint queue are not retained until covered by write-to-disk operations;

maintaining, for each cache, a first checkpoint record for the locally-dirty checkpoint queue ~~that indicates~~ indicating a first time, where all changes made to data items that are presently dirty in the cache prior to the first time have been recorded on a version of the data item that is on persistent storage;

maintaining, for each cache, a second checkpoint record for the globally-dirty checkpoint queue, wherein the second checkpoint record includes a list of data items that were once dirtied in the cache but have since been transferred out and not written to persistent storage; and

after said failure, determining where to begin processing the redo log associated with each cache based on the first checkpoint record and said second checkpoint record for said cache.

36. (Original) The computer-readable medium of Claim 35 wherein the step of processing the redo log comprises the steps of:

determining a starting position for scanning the redo log based on a lesser of a position in the redo log as determined by the first checkpoint record and the positions in the log as determined by the earliest change made to the list of the data items in the second checkpoint record; and

during recovery, for the portion of the redo log between the position indicated by the global checkpoint record to the position indicated by the local checkpoint record, considering for potential redo only those log records that correspond to the data items identified in the global checkpoint record.